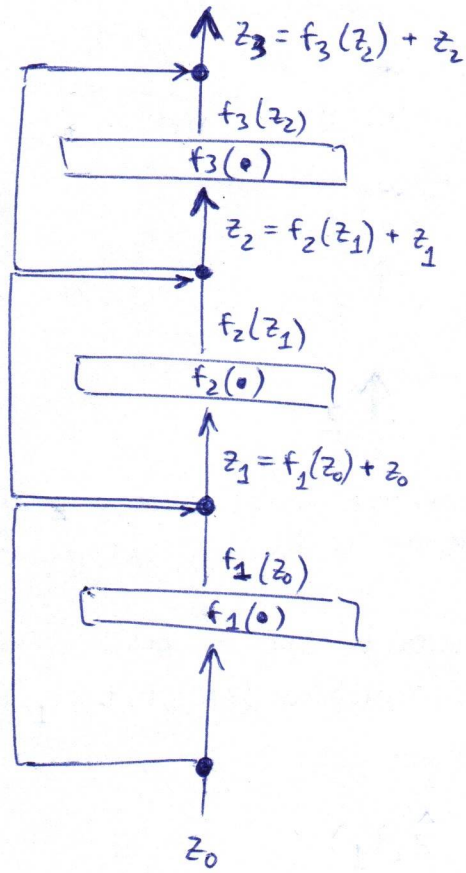
VIDEO "Neural Ordinary Differential Equations" – by "Andriy Drozdyuk"

ResNET



$$z_{i+1} = f_{i+1}(z_i) + z_i$$

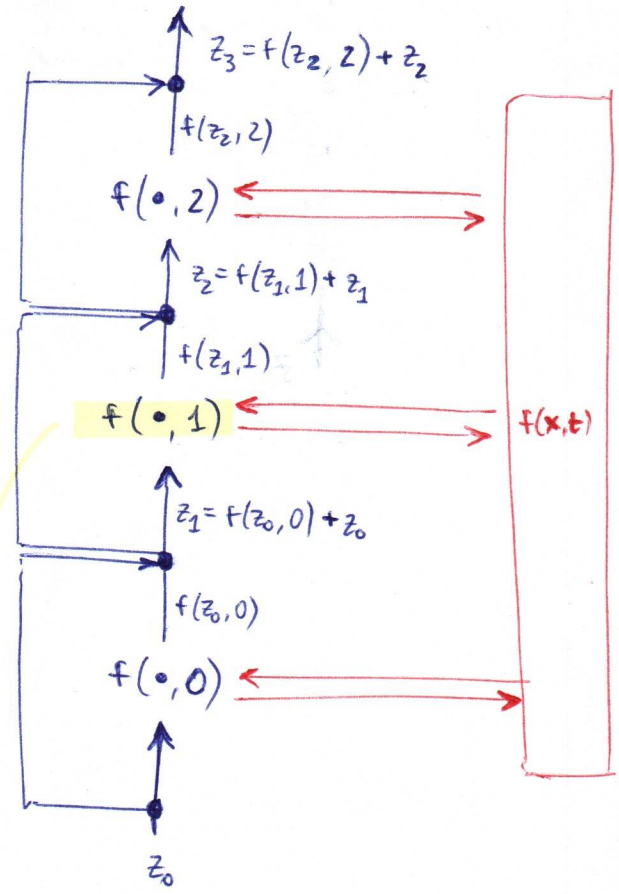
residual — skip connection

the output of every layer is something above to the input $(f_\cdot(z_\cdot))$ + the original input $(z_\cdot)$.

(Note: f_i can be anything and $f_i = \text{or} \neq f_j$ for $i \neq j$)

VS.

ODE Net



what time does this function visit $f(x,t)$? $t = 1$

who brings with him? $\cdot = z_1$

We can see z as a function of time: $z(t)$:

$z_0 = z(0)$
$z_1 = z(1)$
$z_2 = z(2)$
$z_3 = z(3)$

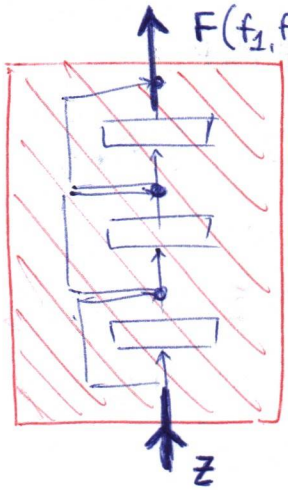
However, we don't know z, it's hidden

$f_i(\cdot)$ | $f(\cdot, t)$

dependence based on which moment we use it

(ResNet)

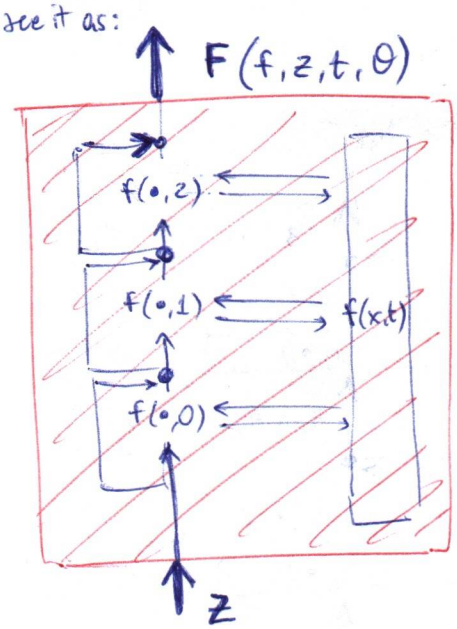
We can see it as:



(supposing that f_i depends on θ)

(ODENet)

We can see it as:



If we know the initial value of z ($z(0)$) we can move to the field of dynamical systems:

$\Rightarrow$ through ODE solvers

(e.g. ODESolve $(z(0), f, t_0, t_1, \theta)$)

we get:

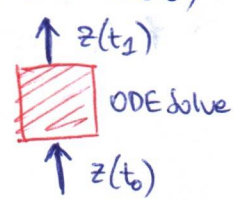
$$\hat{z}(t_1)$$

$\Rightarrow$ the output of the neural network will be the (approximated) evaluation of $z(\cdot)$ at t_1.

We define the loss function as:

$$\hat{z}(t_1) - z(t_1) .$$

Basically, through some black box method (ODE solver) we get the output $z(t_1)$ starting from $z(t_0)$



Once we have $\hat{z}(t_1)$ we evaluate the loss. How do we backpropagate the error through ▨ ?

→ ADJOINT METHOD

Dynamical Systems:

characterized by:

- a state that changes in time:

$$z(t)$$

- a rule for how the state changes:

$$\frac{dz}{dt} = f(t, z, p) \quad \text{ODE}$$

← usually this is a system of ODEs

t: time, z: state, p: parameters

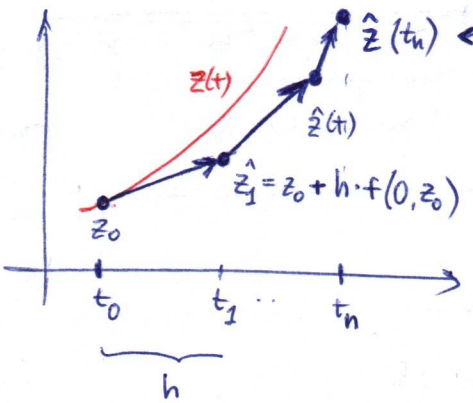
The solution to an ODE is a function: $z(t)$.
We are usually given with an initial value $z(0)$ (more generically: $z(t_0)$).

Finding a solution to a dynamical system:

Given a dynamical system $f(t, z, p)$ with an initial condition z_0, find the solution $z(t)$.

Euler method:

$$z_{t+1} = z_t + h\, f(t, z_t)$$

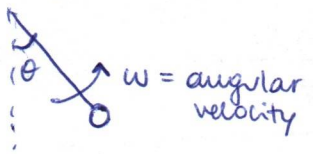


← we have solved the system for this time

$(t_0 = 0)$

the divergence of $\hat{z}(t)$ and $z(t)$ is quite a bit!
However it's still something: we can find a solution $z(t)$ for any t.

Example: Pendulum (pseudopython)



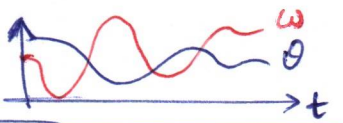
rule of changing the state

```
def pendulum(y, t, b, c):
    θ, ω = y          ← state
    dy/dt = [ω, -bω - c sin(θ)]
    return dy/dt
```

$y_0 = [\pi - 0.1, 0.0]$ — initially pendulum is nearly vertical

```
t = np.linspace(0, 10, 101)   — 101 points of 0 ≤ t ≤ 10
from scipy.integrate import odeint
sol = odeint(pendulum, y0, t, args=(b, c))
```

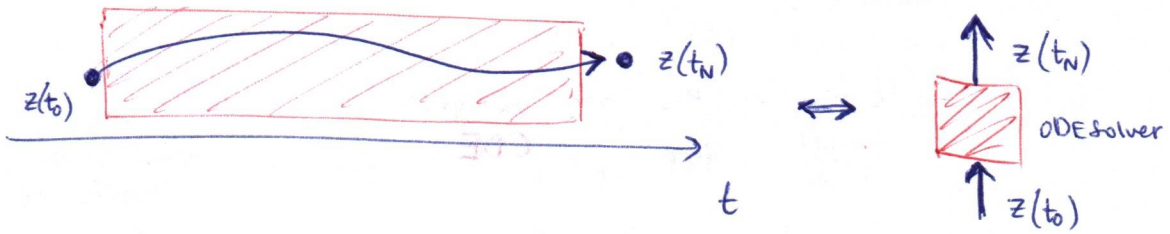
⟹ plot `sol[:,0]`, `sol[:,1]` :



Adjoint method:

we already know that through an ODEsolver we can obtain $z(t_N)$ starting from $z(t_0)$, f (and t_0, t_N, θ)

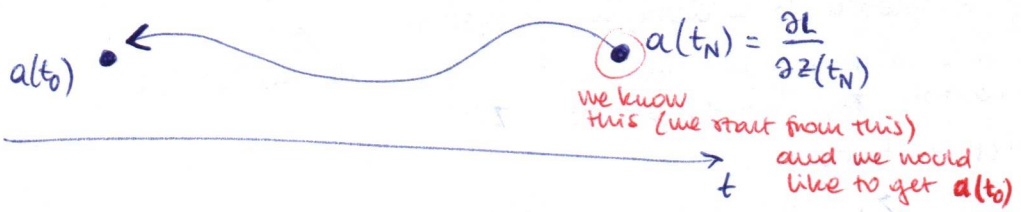
State space:



How can we backpropagate the error?
We define an adjoint.
We define:

$$a(t_i) := \frac{\partial L}{\partial z(t_i)}$$

adjoint state:



→ we can use the same approach as before:
if we treat $a(t_N)$ as initial condition and if we know how $a(\cdot)$ changes w.r.t. time then we can integrate backwards and obtain $a(t_0)$

"continuous-time backpropagation"

ADJOINT METHOD

- Define: $a(t) := \frac{\partial L}{\partial z(t)}$
- Forward: $z(t+1) = z(t) + \int_t^{t+1} f(z(t))\, dt$ (previous) black-box ODEsolver
- Backward: $a(t) = a(t+1) + \int_{t+1}^{t} a(t) \frac{\partial f(z(t))}{\partial z(t)}\, dt$ (1) — $a(t)$: adjoint state
- Params: $\frac{\partial L}{\partial \theta} = \int_t^{t+1} a(t) \frac{\partial f(z(t), \theta)}{\partial \theta}\, dt$ (2)

How can we obtain adjoint differential equation (1) and gradients w.r.t. θ (2)?

(Adjoint method – pt. 2)

Let $z(t)$ follow the differential equation: $\frac{dz(t)}{dt} = f(z(t), t, \theta)$, where θ are the parameters.
If we define the adjoint state:

$$a(t) := \frac{\partial L}{\partial z(t)}$$

⟶ the adjoint state $a(t)$ follows the diff. equation:

$$\frac{da(t)}{dt} = -a(t) \cdot \frac{\partial f(z(t), t, \theta)}{\partial z(t)}$$

adjoint dynamics

To really conclude we also need to know how θ vary in time.
Let's say that all $[z, \theta, t]$ changes in time.
We call $[z, \theta, t]$ augmented state.

$$\underbrace{\frac{d}{dt}\begin{bmatrix} z \\ \theta \\ t \end{bmatrix}(t)}_{\text{derivative of the augmented state}} = f_{aug}([z, \theta, t]) := \begin{bmatrix} f([z, t, \theta]) \\ 0 \\ 1 \end{bmatrix} \begin{matrix} \\ \rightarrow (\text{Hp. } \frac{d\theta}{dt} = 0) \\ \rightarrow (\frac{dt}{dt} = 1) \end{matrix}$$

We define the augmented adjoint as:

$$a_{aug} = \begin{bmatrix} a \\ a_\theta \\ a_t \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z(t)} \\ \frac{\partial L}{\partial \theta(t)} \\ \frac{\partial L}{\partial t(t)} \end{bmatrix}$$

$$\longrightarrow \frac{d a_{aug}(t)}{dt} = -\left[a \frac{\partial f}{\partial z}, \; a \frac{\partial f}{\partial \theta}, \; a \frac{\partial f}{\partial t} \right](t)$$

⟶ Algorithm 1 (Reverse-mode derivative of an ODE initial value problem)
(paper "Neural Ordinary Differential Equations")